Naval Research Laboratory
Stennis Space Center, MS 39529-5004

NRL/FR/7441--92-9417

Color Reproduction Based on Red, Green, and Blue Primaries for a Cyan-, Magenta-, and Yellow-Based Hardcopy Device

STEPHANIE A. MYRICK
MAURA C. LOHRENZ
PERRY B. WISCHOW

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

August 3, 1993

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 3, 1993	Final

4. TITLE AND SUBTITLE

Color Reproduction Based on Red, Green, and Blue Primaries for a Cyan-, Magenta-, and Yellow-Based Hardcopy Device

5. FUNDING NUMBERS

Job Order No. 574526903
Program Element No. APN
Project No.
Task No.
Accession No. DN257017

6. AUTHOR(S)

Stephanie A. Myrick, Maura C. Lohnrez, and Perry B. Wischow

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/7441--92-9417

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Air Systems Command
Department of the Navy (AIR-803)
Washington, DC 20361-8030

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report presents a procedure that is used to reproduce custom colors on color hardcopy. The set of custom colors is comprised of red, green and blue (RGB) intensities, where intensity levels range from 0 (no intensity) to 255 (maximum intensity). The hardcopy device that produced the results in this report normally transforms RGB colors into cyan, magenta, yellow, and black. However, adding black ink during the printing process tends to cause the loss of some low-intensity colors and an overall graying of the output image. This phenomenon has been termed "color drop-out." The procedure described eliminates color drop-out in custom color reproduction by omitting black ink. This approach more accurately reproduced custom colors for the data set used in this study. Two other less successful methods are also presented.

14. SUBJECT TERMS

Mapping, charting, color reproduction, compressed aeronautical chart

15. NUMBER OF PAGES
24

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Same as report

CONTENTS

COLOR REPRODUCTION BASED ON RED, GREEN, AND BLUE PRIMARIES FOR A CYAN-, MAGENTA-, AND YELLOW-BASED HARDCOPY DEVICE

INTRODUCTION

The Navy is developing a database of scanned aeronautical chart images, the Compressed Aeronautical Chart (CAC), for use in aircraft digital moving-map systems and for mission planning [1]. The CAC uses a set of 30 custom color palettes. Each palette consists of 240 distinct colors, and each color is comprised of red-green-blue (RGB) intensities. Intensity levels range from 0 (no intensity) to 255 (maximum intensity). Although the CAC is primarily destined for video display, the need for high-resolution color hardcopies also exists. Available plotter hardware and software produced unacceptable colors, since the low intensities of certain CAC palette colors resulted in poor color reproduction in hardcopy. Low-intensity palette colors suffered from color drop-out, which tends to produce colors with a gray-shade appearance. It was discovered that color drop-out is caused by the addition of black ink, which is specified by the plotter software. Some color hardcopy devices produce black by blending the three primary inks (RGB) or cyan, magenta, yellow (CMY). However, this process often results in a somewhat muddy black. This problem has been solved for other devices by using black ink to produce true black. These devices also add black ink to colors other than true black, which, when combined with the subsequent reduction in the CMY intensities for those colors, can produce the graying effect of color drop-out.

This report presents a procedure that eliminates color drop-out and reproduces custom CAC palette colors that are comparable in quality to the colors found in original aeronautical charts.

PROBLEM

The custom color palettes for CAC data are based on intensities of the additive primary colors: RGB. The in-house, high-resolution, color plotter hardware, which relies on the complementary subtractive primary colors, CMY, produced plots with unacceptable colors. In particular, the lower intensity colors suffered from color drop-out, which resulted in a gray-shade appearance (Fig. 1). Higher intensity colors suffered less from color drop-out but still did not reproduce well.

CAC RGB intensities are normalized for use with conventional display devices by using the following algorithm:

$$\begin{aligned} R &= R/255.0 \\ G &= G/255.0 \\ B &= B/255.0 \end{aligned} \qquad (1)$$

The resulting RGB values range from 0.0 to 1.0 instead of 0 to 255.

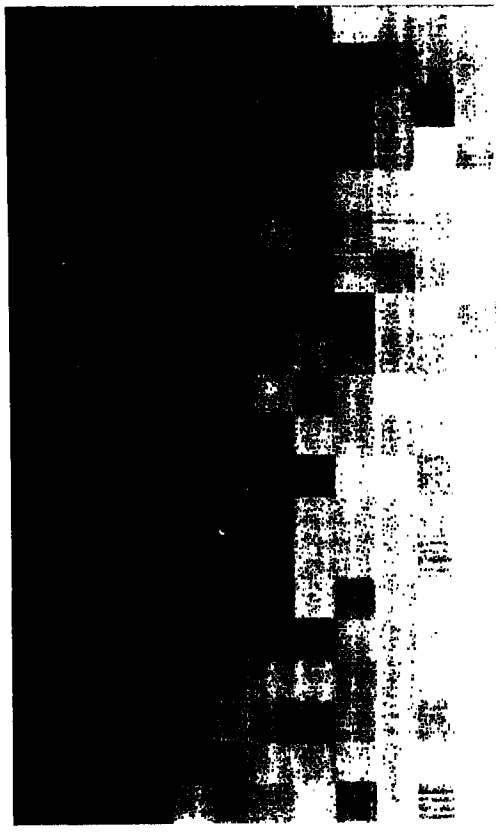

Fig. 1 — CAC custom color palette with color drop-out.

The in-house plotter plots CAC data by first converting the normalized CAC RGB intensities to their CMY equivalents. The following transformation [2] converts from RGB to CMY:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2}$$

CMY values then are adjusted to reflect the plotter's addition of black ink. The adjustment algorithm [2] uses black (K) in place of equal amounts of CMY as follows:

$$\begin{aligned} K &= \text{minimum } (C,M,Y) \\ C &= C - K \\ M &= M - K \\ Y &= Y - K \end{aligned} \tag{3}$$

Using a 4×4 pixel pattern, CMYK inks are deposited on paper as a grid of colored dots. The orientation of each colored dot allows the eye to spatially integrate light that is reflected from adjacent dots. For any given CMYK component, a 7% minimum intensity is required to have at

least 1 element activated, out of 16 in the pixel matrix. CMYK components with intensities of less than 7% (i.e., no color) will not be plotted. The following algorithm [3] is used to calculate the number of elements (within the matrix) to activate for a given CMYK intensity:

$$\text{Number of elements} = (\text{intensity} * \text{matrix size}) / 100 , \quad (4)$$

where intensity= (intensity × 100) + 0.5, and matrix size = 4 × 4 elements.

The CAC palette has been sorted by increasing intensity. As shown in Fig. 1, color drop-out was most severe in the lower intensities. One attempt to alleviate color drop-out was based on the hue, lightness, saturation color model [2] in which the CAC color palette was shifted toward white along the achromatic axis to increase the overall color palette intensities (Fig. 2). This achromatic shift succeeded in producing less color drop-out in lower intensity colors. However, the shift resulted in the opposite problem: lighter intensity colors were "washed out" (i.e., too much white).

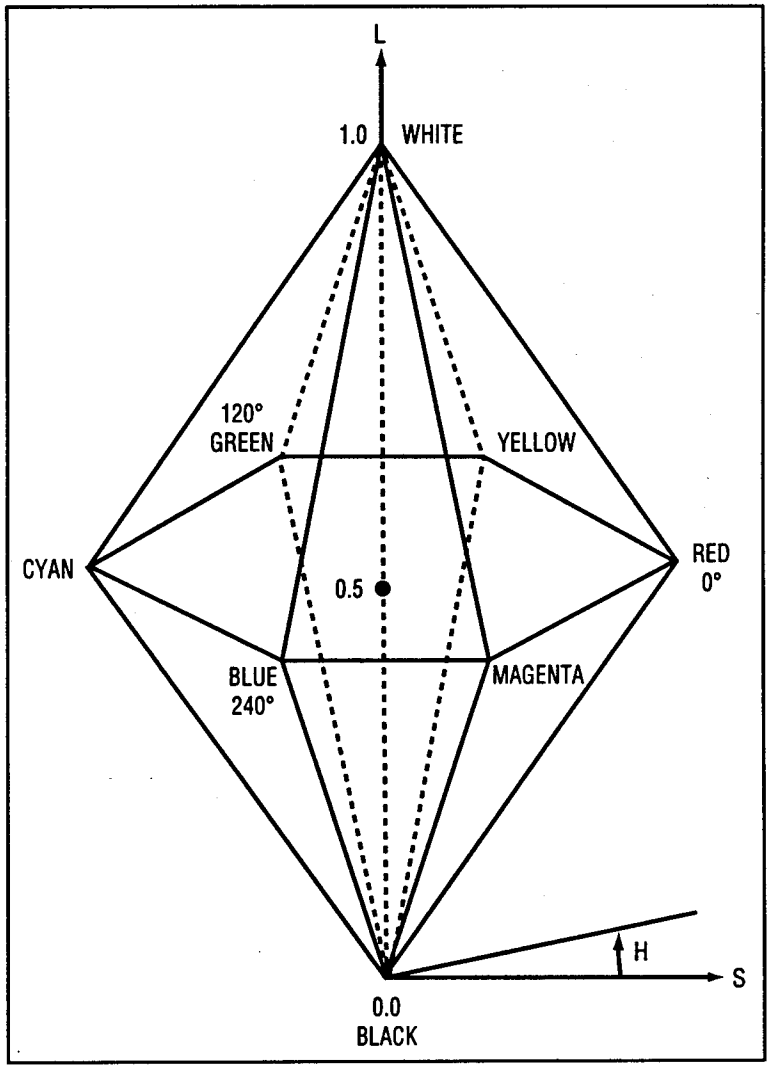

Fig. 2 — HLS color model

Applying a combination of achromatic shifts to individual colors was deemed undesirable due to the significant number of colors that would have to be modified (240 colors in each of the 30 standard CAC color palettes). Even with the availability of examples of computer-generated colors [4], where percentages of each CMYK ink are provided, there would be simply too many individual CAC palette colors to be matched.

Another attempt to alleviate color drop-out involved increasing the pixel pattern to 8×8. This procedure required a minimum intensity of 2% for any given CMYK color to have at least 1 element activated out of 64. Although the resulting plots had less color drop-out, this attempt proved to be unsatisfactory because they suffered a gridded appearance due to the orientation of colored dots within the matrix.

An underlying problem with both of these approaches is the large number of colors to be manipulated. Rather than working with every color in a given palette, a test palette was devised. Since CAC data is comprised of RGB intensities (which are later transformed into their CMYK components), various intensities of RGB were selected as test colors. The intensities of the colors were chosen to vary in lightness by 12%, from little to full pure color. Figure 3 presents a plot of these colors and shows how adding black ink influences the appearance of the resultant RGB colors. Except in the case of colors with full intensity (e.g., pure red, pure green, or pure blue), black ink is always noticeably present.

SOLUTION

The CAC color palettes contain true black, but true black is rarely found in the digitized aeronautical charts. It is present in CAC palettes primarily for areas of no data coverage. Since the CAC data rarely (if ever) require true black, and since the addition of black by the plotter seriously compromises the quality of many other CAC colors, the final solution to the color drop-out problem pivoted around the elimination of black ink from the plotting process.

The FORTRAN source code for the plotter software [3] was modified in-house to completely omit the inclusion of black ink: RGB intensities were converted to their CMY equivalents (Eq. 2), but the adjustment algorithm (Eq. 3) for black ink was eliminated. Figure 4 presents the resulting test plot of RGB colors (in which intensity varies by 12% from little to pure color) using CMY inks and no black ink. The lower intensity RGB colors, which previously suffered from color drop-out, now are acceptable in appearance. The higher intensity RGB colors are also acceptable. Using the modified plotter software, another custom color palette plot was created. As shown in Fig. 5, the palette suffered no appreciable color drop-out, and all colors were more accurately reproduced.

CONCLUSIONS AND RECOMMENDATIONS

Results from the color experiments documented here indicate that the addition of black ink interferes with the reproduction of custom colors on some hardcopy plotters. The Navy standard CAC database was used in this study. CAC data use a standard set of 30 custom-designed color palettes, each of which consists of 240 RGB colors. The custom RGB colors in CAC data did not reproduce well on an in-house, CMYK-based, color hardcopy device that added black ink during the RGB to CMYK transformation. In particular, the addition of black ink resulted in a graying effect, termed color drop-out, in most CAC colors. Several adjustments were considered, including

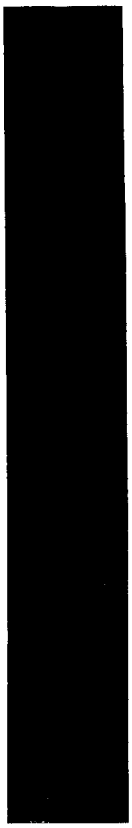

Fig. 3 — RGB test plot using CMYK inks.

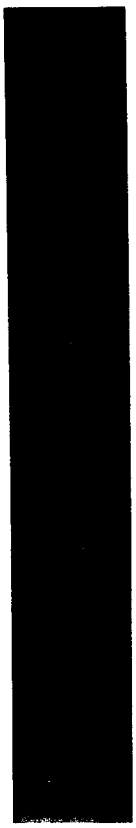

Fig. 4 — RGB test plot using CMY inks and no black (k) ink.

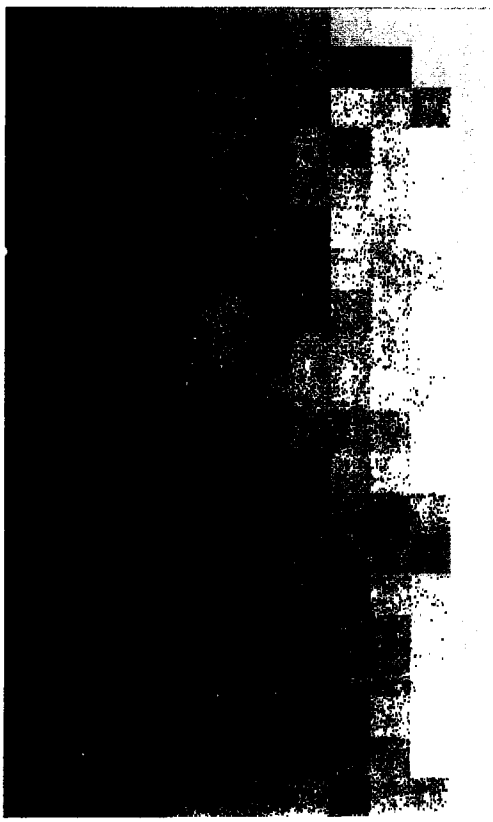

Fig. 5 — CAC custom color palette without color drop-out.

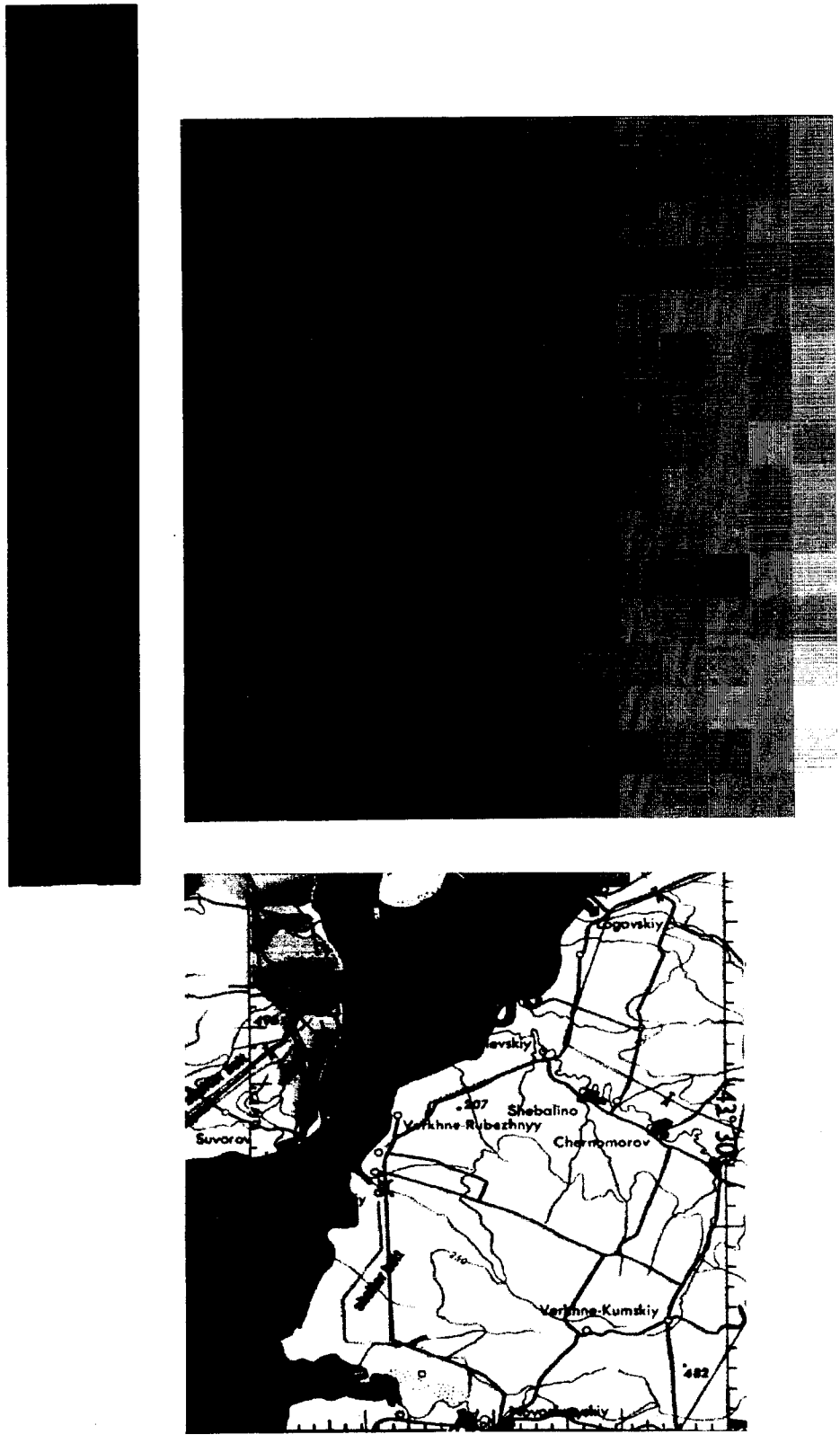

Fig. 6 — Reference plot with CAC data.

a shift of the colors along the achromatic axis (toward higher overall intensities), increasing the size of the pixel pattern matrix from 4×4 to 8×8 pixels, and eliminating the inclusion of black ink. The first two approaches did not adequately solve the color reproduction problems; the third approach proved to be the best solution. Since true black is rarely needed by the CAC database, and since black can be approximated by blending CMY colors, the use of black ink was completely eliminated. Test plots revealed that the deletion of black ink from the RGB to CMYK transformation resulted in the successful reproduction of all CAC custom colors. Appendix A contains source code for the new program, DISPLAY_CAC_CALCOMP.FOR, that is used to plot CAC data on the electrostatic plotter.

An interesting corollary to this study was the demonstrated vulnerability of plot quality to environmental and chemical factors. The quality of plot appearance was profoundly influenced by environmental factors, such as room temperature and humidity. Chemical factors relating to the toner and replenisher (i.e., age and percentage used) also affected plot quality. The adverse influence of these factors was manifested by uneven distribution and absorption of ink. Due to the significant influence of environmental and chemical factors on plot quality and appearance, reference plots are now included with all in-house plotter output. These reference plots, which include smaller versions of the RGB and color palette plots, help to identify whether a poor plot was due to color problems or to environmental/chemical influences. A typical reference plot, along with a portion of CAC data, is shown in Fig. 6. Appendix B lists source code for the subroutine, REFERENCE_PLOT.FOR, that is used to generate reference plots.

ACKNOWLEDGMENTS

This work resulted from ongoing efforts within the Naval Research Laboratory's Map Data Formatting Facility (MDFF). This particular MDFF project was funded by the Naval Air Systems Command F/A-18 (program element APN) and V-22 (program element 0604262) programs. The authors thank the following program managers at NAVAIR for their support: CDR V. J. Chenevy, LCDR C. Cleaver, and Major J. P. Stevens (F/A-18); and CDR M. Redshaw (V-22).

REFERENCES

1. M.C. Lohrenz and J.E. Ryan, "The Navy Standard Compressed Aeronautical Chart Database," Naval Research Laboratory, Stennis Space Center, MS. NOARL Report 8.

2. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice* (Addison-Wesley, Reading, MA, 1990). p. 588–594.

3. COLRGN, CalComp Inc., Anaheim, CA.

4. M. Rogondino and P. Rogondino, *Computer Color: 10,000 Computer-Generated Process Colors* (Chronicle Books, San Francisco, CA, 1990).

APPENDIX A

```
C*************************************************************************
C
C       TITLE:  Display_CAC_CalComp.FOR
C
C DESCRIPTION: This program creates a plot file of Compressed
C              Aeronautical Chart (CAC) data.  Specified rows and
C              columns of CAC segments are decompressed and written
C             to a CalComp-formatted file.  This file contains the required
C              format for plotting CAC segments on a CalComp electrostatic
C              color plotter.
C
C              The CAC is produced by the Naval Research Laboratory (NRL),
C              Stennis Space Center, Mississippi.  Some familiarity
C              with CAC is required to execute this program.
C              See NOARL Report 8, July 1990 (product specification)
C              for additional information about CAC.
C
C              Reference plots which include a Red, Green, Blue color scale
C              and a CAC color palette are optional.
C
C      OUTPUT: A binary plot file (in CalComp format) containing CAC
C             coverage over the specified rows and columns is written in the
C              user's current directory.  The file name is supplied as
C              program input.
C
C      AUTHOR: Stephanie Myrick
C              NRL Code 7441
C              Stennis Space Center, MS 39529-5004
C
C*************************************************************************

      program DISPLAY_CAC_CALCOMP

     implicit none

      include '($RMSDEF)'       ! VAX/VMS System Definition

      byte      rgbtable(256,3)      ! Color table extracted from color
                                     !    "palette.dat"
      byte      seg_buffer(256,256) ! Uncompressed segment
      byte iktmp(4)                  ! Temp variable equivalenced with KITMP
```

```
character*80    plotfilename    ! Output CalComp plot file name
character*80    filename        ! Compressed segment file name
character*132   palname         ! Full color palette file name
character*132   wild_card       ! Wild card file specification
character*132   pa_filename     ! PA directory as a file name
character*80    pa_dir          ! Palette directory path
character*80    pa_found        ! Palette number found
character*1     scale_char      ! Chart scale
character*4     pa_char         ! Palette number
character*6     row_char        ! Row number, user input
character*6     col_char        ! Column number, user input
character*1     reply           ! User reply, input

integer*4       length          ! Length of plotfilename
integer*4       seg_plot(256,256) ! Compressed segment plot buffer
integer*4       kitmp           ! Temp variable equiv. with IKTMP
integer*4       row             ! Current row
integer*4       col             ! Current column
integer*4       start_row       ! Start row of display
integer*4       start_col       ! Start col of display
integer*4       end_row         ! Last row of display
integer*4       num_rows        ! Number of row to display
integer*4       num_cols        ! Number of columns to display
integer*4       mark1,mark2     ! Denotes substring locations
integer*4       pa_dir_len      ! Length of PA directory name
integer*4       pa_found_len    ! Length of palette name
integer*4       pal_color       ! Color palette index
integer*4       palette_offset  ! Offset for loading our color palette
integer*4       pal_shift       ! Palette shifting
integer*4       ldev            ! Logical device number
integer*4       file_stat       ! File open status
integer*4       file_len        ! Length of compressed segment file
integer*4       context,status  ! Used w/RMS calls to locate files
integer*4       input_stat      ! Status of user input
integer*4       isize           ! COLRGN argument, matrix size
integer*4       zone            ! M4 zone
integer*4       i,j             ! Temporary array indeces
integer*4       DECOMPRESS_SEGMENT ! Routine to open & read
                                   ! a compressed segment
integer*4       LIB$FIND_FILE   ! RMS routine to locate a file

logical*1       want_ref        ! Indicates use of RGB and
                                ! color palette reference plots
logical*1       NO_MORE_FILES   ! RMS file search

   real*4       red,grn,blu     ! RED GREEN BLUE colors
   real*4       x_axis,y_axis   ! x- and y-axis plot origins
   real*4       x_position      ! x-axis plotting position
   real*4       y_position      ! y-axis plotting position
   real*4       ifactor         ! plot expansion factor
   real*4       ifactor_offset  ! factor dependent offset for plots

  equivalence (kitmp,iktmp)
```

```
C*********************
C*  Initializations  *
C*********************

        ldev   = 9        ! Output file logical device number
        isize  = 4        ! Use this as the default size
        x_axis = 0.0      ! x axis origin
        y_axis = 0.0      ! y axis origin
        x_position = 0.0 ! initial x-axis plot position
        y_position = 0.0 ! initial y-axis plot position

        call INIT_MEM     ! Initialize memory for CAC utility software

C**** Open output plot file ****
        write (6,*) ' '
        write (6,*) 'Enter the output CalComp plot file name'
        write (6,*) '(Omit a file extension)'
        write (6,*) ' '
        read  (5,'(a)') plotfilename

        call STRING_LENGTH (plotfilename,length)

        plotfilename(length+1:length+4) = '.DAT'  ! the default extension

        open  (ldev,file=plotfilename,status='new',
     &         recordtype='variable', blocksize=484,err=9001)

        write (6,*) ' '
        write(6,*) 'Opened CalComp plot file: ',plotfilename(1:length+4)


C**** Calcomp call: plot initialization ****
        call plots(x_axis,y_axis,ldev)


C**** returns here via ^Z to re-enter program input ****
100  write (6,*)
        input_stat=-1
        do while (input_stat. ne. 0)
           write(6,'('' Enter SCALE (CTRL_Z to quit):  '',$)')
           read(5,'(a)',iostat=input_stat,end=9000) scale_char
        enddo

        write (6,*)
        input_stat=-1
        do while (input_stat .ne. 0)
           write(6,'('' Enter PA#: '',$)')
           read(5,'(a)',iostat=input_stat,end=9000) pa_char
        enddo

        write (6,*)
        input_stat=-1
        do while (input_stat .ne. 0)
           write(6,'('' Enter ZONE (0,1,2,3,4): '',$)')
           read(5,*,iostat=input_stat,end=9000) zone
        enddo
```

```
        write (6,*) ' '
        input_stat=-1
        do while (input_stat .ne. 0)
           write(6,*)'Do you want to include the RGB and'
           write(6,'('' color palette reference plots? (Y/N) '',$)')
           read(5,'(a)',iostat=input_stat,end=9000) reply
        enddo

C**** By default, the reference plots are included ****
        if ((reply .eq. 'Y') .or. (reply .eq. 'y') .or.
     &      (reply .eq. ' ') ) want_ref = .true.

C**** you need to set IFACTOR depending on the isize selected ****
C**** plotting offset will not change                         ****
        ifactor = 0.5
        ifactor_offset = 5.12

C**** construct directory path ****
        pa_dir = 'CHART_ODI_DISK:[MAP' // scale_char // ']'
        call STRING_LENGTH (pa_dir,pa_dir_len)

        pa_found = ' '
        pa_filename = ' '
        context = 0
        wild_card = '*.*;*'
        NO_MORE_FILES = .false.

C**** search for palette directories ****
        do while ((pa_char .ne. pa_found(4:6))  ! palette #'s don't match
     &       .or. (NO_MORE_FILES .eq. .false.))

            status = LIB$FIND_FILE(( pa_dir(1:pa_dir_len) //
     &                 'pa*.dir;*'),
     &                 pa_filename, context, wild_card)
            if (status .eq. RMS$_NMF) then
                continue
            else if (status .eq. RMS$_NMF) then
                  NO_MORE_FILES = .true.
            else
              NO_MORE_FILES = .true.
            end if

            MARK1 = index ( pa_filename, ']')
            MARK2 = index ( pa_filename, '.DIR')
            pa_found = pa_filename(MARK1+1:MARK2-1)

        enddo
```

```
C**** Read the color palette and separate the RGB values      ****
C**** for loading the color table into the graphics processor ****

        call STRING_LENGTH (pa_found,pa_found_len)
                  pa_dir =  pa_dir(1:pa_dir_len - 1)// '.' //
     &            pa_found(1:pa_found_len) //
     &            ']'
        call STRING_LENGTH (pa_dir, pa_dir_len)

C**** READ_PALETTE is a C function, hence character string ****
C**** must be terminated with a null character              ****

        palname = (pa_dir(1:pa_dir_len)//'palette.dat' // char(0) )

        call READ_PALETTE (%val(0),%ref(palname),
     &                     rgbtable(1,1),
     &                     rgbtable(1,2),
     &                     rgbtable(1,3))

C**** Load the color palette: RED, GREEN, BLUE                          ****
C**** An offset of 250 is required to skip over memory locations        ****
C**** that are reserved for CalComp pre-defined symbols in use          ****
C**** when isize = 4.  We will use an offset of 255                     ****
        palette_offset =  255   !  255 for isize = 4
                                !    0 for isize = 8
        do i = 1,240

           IKTMP(1) = rgbtable(i,1)
           red = kitmp                  ! used to be + pal_shift
           red = red/255.0              ! remove declaration if OK

           iktmp(1) = rgbtable(i,2)
           grn = kitmp
           grn = grn/255.0

           iktmp(1) = rgbtable(i,3)
           blu = kitmp
           blu = blu/255.0

C          **** Calcomp call: generate a user-defined color ****
           call colrgn(2,red,grn,blu,i+palette_offset,isize)

        enddo


C**** Determine ROW/COL values of data in specified directory ****
C**** Return here via ^Z to re-enter ROW/COL info             ****
 150    write (6,*)
        input_stat=-1
        do while (input_stat .ne. 0)
          write(6,'('' Enter start ROW  (CTRL_Z to change
```

```
     &     ROW/COL): '',$)')
           read(*,'(a)',iostat=input_stat,end=150) row_char
           if (row_char .ne. ' ') then
               read (row_char,'(bn,i6)') start_row
           end if
         enddo

         write (6,*)
         input_stat=-1
         do while (input_stat .ne. 0)
           write(6,'('' Enter start COL  (CTRL_Z to change
     &             ROW/COL): '',$)')
           read(*,'(a)',iostat=input_stat,end=150) col_char
           if (col_char .ne. ' ') then
               read (col_char,'(bn,i6)') start_col
           end if
         enddo

         write (6,*)
         input_stat=-1
         do while (input_stat .ne. 0)
            write(6,'('' Enter the number of ROWs to plot
     &          (CTRL_Z to change ROW/COL): '',$)')
           read(*,'(a)',iostat=input_stat,end=150) row_char
           if (row_char .ne. ' ') then
               read (row_char,'(bn,i6)') num_rows
           end if
         enddo

         write (6,*)
         input_stat=-1
         do while (input_stat .ne. 0)
            write(6,'('' Enter the number of COLs to plot
     &          (CTRL_Z to change ROW/COL): '',$)')
           read(*,'(a)',iostat=input_stat,end=150) col_char
           if (col_char .ne. ' ') then
               read (col_char,'(bn,i6)') num_cols
           end if
         enddo

C         call draw_infobox (palname, scale_char, zone, start_row,
C     *                           start_col, num_rows, num_cols)

C**** If desired, display RGB and color palette reference plots ****
        if (want_ref) then
           call reference_plot
           x_position = 5.0   ! Reference plot has used approx. 5 inches,
C          ***** CalComp call: adjust X and Y origins *****
           call plot (x_position,y_position,-3)
         endif                    ! adjust x-axis
```

```
C***** CalComp call: expand plot by this factor *****
      call factor(ifactor)


C***** Need to replace the directory terminator ']' with *****
C***** a subdirectory '.' in PA_DIR for appending Row    *****
C***** subdirecotry specification .Rsnnnnn]              *****
       call STRING_LENGTH(pa_dir,pa_dir_len)
       pa_dir = pa_dir(1:pa_dir_len-1) // '.' // char(0)

C**** Display compressed chart data plots with the upper right segment ****

       end_row = start_row + num_rows -1

       do row = end_row,start_row,-1    ! each iteration plots a row

       do col = start_col,start_col+num_cols-1 ! each iteration plots a col

C            ***** Build file name:                                    *****
C            ***** GET_SEGMENT_NAME is a C function. Hence character *****
C            ***** strings must be terminated with a null character  *****
C            ***** and arguments must be passed by value & reference *****

            call GET_SEGMENT_NAME(%ref(pa_dir),%val(row),
     &                            %val(col),%val(zone),
     &                            %ref(filename) )

          call STRING_LENGTH (filename,file_len)

C            **** Read and decompress a segment file. If no ****
C            **** file exists, a "blank" segment (i.e. a    ****
C            **** hole) in the plot results                 ****
          file_stat = DECOMPRESS_SEGMENT (
     *                %ref(filename),
     *                seg_buffer)

C            **** Check for irregular return status, normal = 1 ****
          if (file_stat .ne. 1) then
             print*
             print*,'>> No data for ROW',row,'   COL ',col
             print*,'   ',filename(1:file_len), ' NOT found...'
          else
             write(6,*)
             write(6,7000) row, col
7000         format(' Loading data for ROW: ',i6.5,'   COL: ',i6.5)
```

```
C                ***** load data for plotting *****
                 do i=1,256
                   do j=1,256
                      IKTMP(1) = seg_buffer(i,j)
                      seg_plot(i,j) =  KITMP  + palette_offset + 1

C                     ***** Check for colors out of range *****
                      if ((kitmp+palette_offset+1 .lt. 256) .or.
     &                    (Kitmp+palette_offset+1 .gt. 495))
     &                    write(6,*) 'Indexing color out of range',
     &                               kitmp+palette_offset+1

                   enddo
                 enddo

          endif     ! for file_stat = 1


C            ***** Calcomp call: adjust x & y plot positions
             call plot(x_position,y_position,3)
             type *,'plotting x & y',x_position,y_position



C            ***** CalComp call: fill plot buffer *****
             call rasfil(0.02,0.02,256,256,seg_plot)


C            **** adjust y coordinate - move over 250 pixels ****
C            **** for plotting the next segment/column       ****
             y_position = y_position + ifactor_offset

          enddo   ! EndDo columns

C         **** Adjust x coordinate - move up 250 pixels  ****
C         **** for plotting the next row of segments     ****
          x_position = x_position + ifactor_offset

C         **** Adjust y coordinate - reset to origin    ****
C         **** for plotting the next column of segments ****
          y_position = 0.0

     enddo   ! EndDo rows

C       ***** Calcomp call: fill every image pixel *****
C       call rasfil(0.01,0.01,1000,1000,seg_plot)

C       ***** CalComp call: enlarge the size of the entire plot *****
```

```
C       ***** Calcomp call: terminate plot *****
        call plot(0.,0.,999)

        close(ldev)

9000    stop ' End of program DISPLAY_CAC_CALCOMP'

9001    stop ' Error opening CalComp plot file! Aborting the program'

        end
```

APPENDIX B

```
C*************************************************************************
C
C  TITLE: Reference_Plot.FOR
C
C DESCRIPTION: A FORTRAN subroutine which utilizes CalComp software for
C              plotting Red, Green and Blue (RGB) intensities and a
C              CAC custom color palette.
C              20 RGB Intensities, varying by 12.75% simulate the original
C              CAC intensities that range from 0-255.
C
C ARGUMENTS: None
C
C   RETURNS: None
C
C REQUIRED SUBROUTINES:
C
C        LOAD_PLOT_BUF: Fills the plot buffer with 25x25 pixel "blocks"
C                       of color.  The array dimensions of the plot buffer
C                       are adjustable.
C
C    AUTHOR: Stephanie Myrick
C            NRL, Code 351
C            Stennis Space Center,MS 39529-5004
C
C*************************************************************************
      subroutine reference_plot

         implicit none

         integer*4       rgb_buf(500,500) ! Plot buffer
         integer*4       palette_buf(400,400) ! Plot buffer
         integer*4       pal_color        ! Color palette index
         integer*4       num_horiz        ! Number of horizontal color blocks
         integer*4       num_vert         ! Number of vertical color blocks
         integer*4       num_pixels       ! Number of pixels comprising the
                                          ! color block dimensions, x-y plane
         integer*4       rgb_offset       ! Offset for loading RGB values
         integer*4       palette_offset   ! Offset for loading our color
                                          ! palette
         integer*4       isize            ! COLRGN argument, matrix size
         integer*4       i                ! DO loop counter
```

```
        integer*4       x_buf           ! Plot buffer x dimension
        integer*4       y_buf           ! Plot buffer y dimension

        real*4          x_position      ! Plot x-axis position
        real*4          y_position      ! Plot y-axis position
        real*4          red,grn,blu     ! individual RED GREEN BLUE colors
        real*4          rgb             ! RGB intensity
        real*4          intensity       ! incremental intensity for RGB

C***********
C*  Begin  *
C***********

        isize = 4

C**** Create pure RGB colors that vary in 20 intensities   ****
C**** Intensities range from little to pure color          ****
C**** Allow enough offset for reserved locations & palette ****
C**** where 250 locations are reserved for CalComp symbols ****
C**** and 240 locations are reserved for CAC color palette ****

        rgb = 12.75
        rgb_offset = 500    ! memory location offset
        intensity  = 12.75

        do i = 1,19
           red = rgb / 255.0
           grn = rgb / 255.0
           blu = rgb / 255.0

           call colrgn (2,red,0.0,0.0,i+rgb_offset,isize)
           call colrgn (2,0.0,grn,0.0,i+rgb_offset+20,isize)
           call colrgn (2,0.0,0.0,blu,i+rgb_offset+40,isize)

C          **** Increase color intensity by 1/20th ****
           rgb = rgb + intensity

         enddo

C       ***** load pure RGB values                          *****
C       ***** Since 255 = pure color, pass 1 as intensity *****
         call colrgn (2,1.0,0.0,0.0,20+rgb_offset,isize)
         call colrgn (2,0.0,1.0,0.0,40+rgb_offset,isize)
         call colrgn (2,0.0,0.0,1.0,60+rgb_offset,isize)

C**** Fill a 3x20 color matrix with the RGB intensities *****
        num_horiz  = 3  ! Each horizontal is an RGB primary
        num_vert   = 20 ! These contain the intensities
        num_pixels = 25 ! Each block of color is 25x25 pixels wide

        x_buf = 500  ! RGB_buf array x-dimension
```

```
      y_buf = 500  !                y-dimension

      call load_plot_buf (num_horiz,num_vert,
     &                    num_pixels,rgb_offset,rgb_buf,
     &                    x_buf,y_buf)

C**** Initialize x-axis and y-axis positions ****
       x_position = 0.0
       y_position = 0.0
      call plot(x_position,y_position,3)

C**** CalComp call to fill the raster image ****
       call rasfil (0.01,0.01,500,500,rgb_buf)

C**** CalComp call to modify plotting coordinates ****
C**** Need to move over for the next plot         ****
       x_position = 0.0
       y_position = 2.0
      call plot(x_position,y_position,3)

C**** Load the CAC color palette.  An offset of 250 is required   ****
C**** to skip over memory locations that are reserved for CalComp ****
C**** pre-defined symbols in use when isize = 4.  we will use 255 ****
       palette_offset = 255

C**** Fill a 15x16 matrix of 240 CAC palette colors ****
       num_horiz = 15
       num_vert  = 16
       x_buf = 400
       y_buf = 400

      call load_plot_buf (num_horiz,num_vert,
     &                    num_pixels,palette_offset,palette_buf,
     &                    x_buf,y_buf)

C**** CalComp call to fill the raster image ****
       call rasfil (0.01,0.01,400,400,palette_buf)

     return
     end
```

```
C*****************************************************************
C
C TITLE: LOAD_PLOT_BUF
C
C DESCRIPTION: Fills the buffer containing reference plots.
C              Creates a matrix of square color "blocks" which are
C              composed of (NUM_PIXELS x NUM_PIXELS) pixels.
C              Colors beginning at memory location OFFSET
C              are used to fill these "blocks" of color.
C
C  ARGUMENTS:
C             num_horiz   Total number of horizontal color "blocks"
C             num_vert    Total number of vertical color "blocks"
C             num_pixels  Color "block" dimension. # of pixels in x-y plane
C             offset      Memory offset, pointing to first
C                         location storing color
C            plot_buf    Buffer containing reference plots, ajustable array
C             x-buf       Adjustable array x dimension
C             y-buf       Adjustable array y dimension
C
C    RETURNS: None
C
C     AUTHOR: S.Myrick
C*****************************************************************

      subroutine load_plot_buf (num_horiz,num_vert,num_pixels,
     &                          offset,plot_buf, x_buf,y_buf)

      implicit none

      integer*4   num_horiz ! Total number of horizontal color "blocks"
      integer*4   num_vert  ! Total number of veritcal color "blocks"
      integer*4   num_pixels! Dimension of color "block" in x-y planes
      integer*4   offset    ! Memory offset, pointing to first location
                            ! storing color

      integer*4   x_buf     !  plot buf array dimension
      integer*4   y_buf     !  plot buf array dimension
      integer*4   plot_buf(x_buf, y_buf) ! Buffer containing reference plots
      integer*4   horiz     ! Counter of horizontal color blocks
      integer*4   vert      ! Counter of vertical color blocks
      integer*4   row       ! Current pixel row
      integer*4   col       ! Current pixel column
      integer*4   startrow ! Starting row for creating color "block"
      integer*4   startcol ! Starting column for creating color "block"
      integer*4   endrow    ! Last row for creating color "block"
      integer*4   endcol    ! Last column for creating color "block"
      integer*4   color     ! Color used in filling color "block"
```

```
C*********
C* Begin *
C*********
     color    = offset ! - 1 adjust for DO loop incrementation
        startrow = 1
        endrow   = num_pixels
        startcol = 1
        endcol   = num_pixels

C**** Create a pixel image, or "block".                              ****
C**** Each iteration of the outer loop creates NUM_HORIZ horizontal  ****
C**** blocks of color.                                               ****
C**** Each iteration of the inner loop creates NUM_VERT blocks of color.*
C**** Hence, NUM_HORIZ x NUM_VERT blocks of individual color are created*

       do horiz = 1,num_horiz

C         *** Create "blocks" of color. Each iteration fills          ***
C         *** one block that is NUM_PIXELS pixels wide with one color ***
C         *** These blocks are built or placed horizontally.          ***

          do vert = 1,num_vert
               color = color + 1              ! get a color
               do col = startcol,endcol          ! for  colums/pixels

                 do row = startrow,endrow        ! fill  rows/pixels
                   plot_buf(row,col) = color     ! with this color
                 enddo

               enddo

              startcol = startcol + num_pixels  ! move over to fill
              endcol   = endcol + num_pixels    ! another 25 columns of
                                                ! pixels

        enddo ! for filling in a vertical block of color

        startcol = 1              ! restart another set of columns/pixels
        endcol   = num_pixels
        startrow = startrow + num_pixels   ! move over to fill
        endrow   = endrow   + num_pixels   ! another row of pixels

      enddo ! for filling in a horizontal block of color


    return
       end
```

```
C++
C*************************************************************************
C NAME: draw_infobox
C
C PURPOSE: To draw an information box using calcomp calls that contains the
C         palette used, the start row, the start col, the number of rows and
C          cols, the scale, and the zone.
C
C AUTHOR: Joyce Michelle Mehaffey (Planning System Inc.), Nov 92
C
C SUMMARY:
C   parameter list NONE
C
C DESCRIPTION: This subroutine assumes that a plot file has been open before
C              this routine is called.
C
C
C SYSTEM DEPENDENT FUNCTIONS:
C   When linking this program, you must link in the calcomp library
C==
C*************************************************************************

      subroutine draw_infobox (pa_dir, scale_char, zone, srow,
     *                         scol, nrows, ncols)

      implicit none

      include 'v2_dir:m4_constants.inc'


      integer*4       scale,zone,slen
      integer*4       srow, scol
      integer*4       nrows, ncols
      real*8          y_pos, x_pos    ! start position for a plot

      character*80    string1         ! filename for the plot file
      character*80    pa_dir
      character*3     srow_char, nrow_char
      character*3     scol_char, ncol_char
      character*1     scale_char
      character*2     zone_char
      character*23    create_date

      x_pos = 0.0
      y_pos = 9.0

      call plot(x_pos, y_pos, 3)

      call factor (1.0)       ! always make the factor size = 1.0

      call newpen(11)         ! set pen to draw boldface lines
```

```
      call plot( 0.0,16.0,2)  ! draw thick outside border
      call plot( 1.85,16.0,2)
      call plot( 1.85, 9.0,2)
      call plot( 0.0, 9.0,2)
      call plot( 1.0,9.1 ,3)

      call newpen(3)        ! set pen to draw boldface lines

      call lib$date_time (create_date) ! creation date
      call symbol(1.50,9.25,.10,'created:',,90.0,8)
      slen = INDEX (create_date,'.') -1
      call symbol(1.50,10.25,.10,create_date,,90.0,slen)

      call str$upcase(pa_dir,pa_dir)
      call STRING_LENGTH(pa_dir,slen)
      call symbol(0.25,9.25,.12,pa_dir,,90.0,slen)

      call symbol(0.50,9.25,.12,'Scale: ',,90.0,7)
      call STRING_LENGTH(scale_char,slen)
      call symbol(0.50,10.0,.12,scale_char,,90.0,slen)

      call symbol(0.75,9.25,.12,'Zone: ',,90.0,6)
      zone_char = zone_table(zone)
      call STRING_LENGTH(zone_char,slen)
      call symbol(0.75,10.0,.12,zone_char,,90.0,slen)

      write (srow_char,'(i3)') srow
      write (nrow_char,'(i3)') nrows
      string1 = 'Start row:  '//srow_char//'      Number of rows '
     *    //'plotted:    '//nrow_char
      call STRING_LENGTH(string1,slen)
      call symbol(1.0,9.25,.12,string1,,90.0,slen)

      write (scol_char,'(i3)') scol
      write (ncol_char,'(i3)') ncols
      string1 = 'Start col:  '//scol_char//'      Number of cols '
     *    //'plotted:    '//ncol_char
      call STRING_LENGTH(string1,slen)
      call symbol(1.25,9.25,.12,string1,,90.0,slen)

      return
      end
```